CS4102 Algorithms Spring 2020

Warm up

Decode the line below into English

(hint: use Google or Wolfram Alpha)

•• •-•• •• -•- • •- •-• --- •-• •• - •••• -

CS4102 Algorithms Spring 2020

Warm up

Decode the line below into English

(hint: use Google or Wolfram Alpha)





Today's Keywords

- Greedy Algorithms
- Exchange Argument
- Choice Function
- Prefix-free code
- Compression
- Huffman Code

CLRS Readings

• Chapter 16

Homeworks

- HW6 Due Sunday, April 5 @ 11pm
 - Written (use latex)
 - DP and Greedy
- EC1 also due Tuesday, November 5 @ 11pm
 - No office hours for that assignment
- HW4 grades coming later this week

Greedy Algorithms

- Require Optimal Substructure
 - Solution to larger problem contains the solution to a smaller one
 - Only one subproblem to consider!
- Idea:
 - 1. Identify a greedy choice property
 - How to make a choice guaranteed to be included in some optimal solution
 - 2. Repeatedly apply the choice property until no subproblems remain

Exchange argument

- Shows correctness of a greedy algorithm
- Idea:
 - Show exchanging an item from an arbitrary optimal solution with your greedy choice makes the new solution no worse
 - How to show my sandwich is at least as good as yours:
 - Show: "I can remove any item from your sandwich, and it would be no worse by replacing it with the same item from my sandwich"



Sam Morse



 Engineer and artist

Message Encoding

- Problem: need to electronically send a message to two people at a distance.
- Channel for message is binary (either on or off)



How can we do it?

wiggle, wiggle, wiggle like a gypsy queen wiggle, wiggle, wiggle all dressed in green

 Take the message, send it over character-by-character with an encoding

Character	
requency	Encoding
a: 2	0000
d: 2	0001
e: 13	0010
g: 14	0011
i: 8	0100
k: 1	0101
l: 9	0110
n: 3	0111
p: 1	1000
q: 1	1001
r: 2	1010
s: 3	1011
u: 1	1100
w: 6	1101
y: 2	1110
	1 1

10

How efficient is this?

wiggle wiggle wiggle like a gypsy queen wiggle wiggle wiggle all dressed in green

Each character requires 4 bits

 $\ell_c = 4$

Cost of encoding:

$$B(T, \{f_c\}) = \sum_{character c} \ell_c f_c = 68 \cdot 4 = 272$$

Character	
Frequency	Encoding
a: 2	0000
d: 2	0001
e: 13	0010
g: 14	0011
i: 8	0100
k: 1	0101
l: 9	0110
n: 3	0111
p: 1	1000
q: 1	1001
r: 2	1010
s: 3	1011
u: 1	1100
w: 6	1101
y: 2	1110
	1 1

11

How efficient is this?

wiggle wiggle wiggle like a gypsy queen wiggle wiggle wiggle all dressed in green

Each character requires 4 bits

 $\ell_c = 4$

Cost of encoding:

$$B(T, \{f_c\}) = \sum_{character c} \ell_c f_c = 68 \cdot 4 = 272$$

Better Solution: Allow for different characters to have different-size encodings (high frequency → short code)

Character	
Frequency	Encoding
a: 2	0000
d: 2	0001
e: 13	0010
g: 14	0011
i: 8	0100
k: 1	0101
l: 9	0110
n: 3	0111
p: 1	1000
q: 1	1001
r: 2	1010
s: 3	1011
u: 1	1100
w: 6	1101
y: 2	1110

12

More efficient coding



More efficient coding



Morse Code



Problem with Morse Code

International Morse Code

- The length of a dot is one unit.
- 2. A dash is three units.
- 3. The space between parts of the same letter is one unit.
- 4. The space between letters is three units.
- 5. The space between words is seven units.





Problem with Morse Code

International Morse Code

- The length of a dot is one unit.
- 2. A dash is three units.
- 3. The space between parts of the same letter is one unit.
- 4. The space between letters is three units.
- 5. The space between words is seven units.





Ambiguous Decoding

Prefix-Free Code

• A prefix-free code is codeword table T such that for any two characters c_1, c_2 , if $c_1 \neq c_2$ then $code(c_1)$ is not a prefix of $code(c_2)$

g 0 1111011100011010 e 10 w i gg l e l 110 i 1110 w 11110

Binary Trees = Prefix-free Codes

- I can represent any prefix-free code as a binary tree
- I can create a prefix-free code from any binary tree



Goal: Shortest Prefix-Free Encoding

- Input: A set of character frequencies $\{f_c\}$
- Output: A prefix-free code *T* which minimizes

$$B(T, \{f_c\}) = \sum_{character c} \ell_c f_c$$

Goal: Shortest Prefix-Free Encoding

- Input: A set of character frequencies $\{f_c\}$
- Output: A prefix-free code *T* which minimizes

$$B(T, \{f_c\}) = \sum_{character c} \ell_c f_c$$

Greedy Algorithms

- Require Optimal Substructure
 - Solution to larger problem contains the solution to a smaller one
 - Only one subproblem to consider!
- Idea:
 - 1. Identify a greedy choice property
 - How to make a choice guaranteed to be included in some optimal solution
 - 2. Repeatedly apply the choice property until no subproblems remain



• Choose the least frequent pair, combine into a subtree



Subproblem of size n - 1!













Exchange argument

- Shows correctness of a greedy algorithm
- Idea:
 - Show exchanging an item from an arbitrary optimal solution with your greedy choice makes the new solution no worse
 - How to show my sandwich is at least as good as yours:
 - Show: "I can remove any item from your sandwich, and it would be no worse by replacing it with the same item from my sandwich"



- Overview:
 - Show that there is **an** optimal tree in which the least frequent characters are siblings
 - Exchange argument
 - Show that making them siblings and solving the new smaller subproblem <u>results in</u> an optimal solution
 - Proof by contradiction

• First Step: Show any optimal tree is "full" (each node has either 0 or 2 children)



• First Step: Show any optimal tree is "full" (each node has either 0 or 2 children)



Huffman Exchange Argument

- Claim: if c_1, c_2 are the least-frequent characters, then there is an optimal prefix-free code s.t. c_1, c_2 are siblings
 - i.e. codes for c_1, c_2 are the same length and differ only by their last bit

Case 1: Consider some optimal tree T_{opt} . If c_1, c_2 are siblings in this tree, then claim holds



Huffman Exchange Argument

- Claim: if c_1, c_2 are the least-frequent characters, then there is an optimal prefix-free code s.t. c_1, c_2 are siblings
 - i.e. codes for c_1, c_2 are the same length and differ only by their last bit

Case 2: Consider some optimal tree T_{opt} , in which c_1, c_2 are not siblings



Let *a*, *b* be the two characters of lowest depth that are siblings (Why must they exist?)

Case 2: c_1, c_2 are not siblings in T_{opt}

 Claim: the least-frequent characters (c₁, c₂), are siblings in some optimal tree

a, b =lowest-depth siblings

Idea: show that swapping c_1 with a does not increase cost of the tree. Assume: $f_{c1} \leq f_a$



Case 2: c_1, c_2 are not siblings in T_{opt}

 Claim: the least-frequent characters (c₁, c₂), are siblings in some optimal tree

a, b =lowest-depth siblings

Idea: show that swapping c_1 with a does not increase cost of the tree. Assume: $f_{c1} \leq f_a$

$$B(T_{opt}) = C + f_{c1}\ell_{c1} + f_a\ell_a$$

 $B(T') = C + f_{c1}\ell_a + f_a\ell_{c1}$

Case 2: c_1, c_2 are not siblings in T_{opt}

 Claim: the least-frequent characters (c₁, c₂), are siblings in some optimal tree

a, b =lowest-depth siblings

Idea: show that swapping c_1 with a does not increase cost of the tree. Assume: $f_{c1} \leq f_a$



Case 2:Repeat to swap $c_2, b!$

 Claim: the least-frequent characters (c₁, c₂), are siblings in some optimal tree

a, b =lowest-depth siblings

Idea: show that swapping c_2 with b does not increase cost of the tree. Assume: $f_{c2} \leq f_b$



- Overview:
 - Show that there is an optimal tree in which the least frequent characters are siblings
 - Exchange argument
 - Show that making them siblings and solving the new smaller subproblem results in an optimal solution
 - Proof by contradiction

Finishing the Proof

- Show Optimal Substructure
 - Show treating c_1, c_2 as a new "combined" character gives optimal solution

Why does solving this smaller problem:



Give an optimal solution to this?:





• Claim: An optimal solution for F involves finding an optimal solution for F', then adding c_1, c_2 as children to σ

If this is optimal





















